SEEDB: Efficient Data-Driven Visualization Recommendations to Support Visual Analytics

Names: Nitin Kishore Sai Samala (30168428)

Dung Thai (30522685)

Brief Introduction

Visualizing high dimensional datasets can be laborious. Given a new dataset like the **Census Data** or a new question about an existing dataset, an analyst builds various visualizations to get a feel for the data, to find anomalies and outliers, and to identify patterns that might merit further investigation. A visual recommendation like **SeeDB** intelligently explores the space of visualizations, evaluates promising visualizations for trends, and recommends those it deems most "useful" or "interesting".

Relevant Terms

- Database D with a snowflake schema
- Dimension Attributes A, that we would like to group-by in our visualizations
- Measure attributes M, that we would like to aggregate in our visualizations
- F, the set of potential aggregate functions over the measure attribute
- Query Q, to explore a subset of data

We assume that we can group D along any of the dimension attributes A and we can aggregate any of the measure attributes M and the resulting two-column table can be easily visualized via standard visualization mechanisms, such as bar charts or trend lines. The goal of SEEDB is to recommend visualizations of Q that have high utility. The class of queries Q posed over D that we support encompasses a general class of queries that select a horizontal fragment of the fact table and one or more-dimension tables. We can view this as a simple selection query over the result of joining all the tables involved in the snowflake schema

	marital_status text	count bigint
1	Married-AF-spouse	37
2	Never-married	16116
3	Married-civ-spouse	22379
4	Married-spouse-absent	628
5	Widowed	1518
6	Separated	1530
7	Divorced	6633

We grouped **[Married-AF-spouse, Married-civ-spouse, Married-spouse-absent, Separated]** as married group and **[Never-married, Divorced, Widowed]** as the unmarried group which is our reference data. Both the databases have almost equal size

Measurement attributes

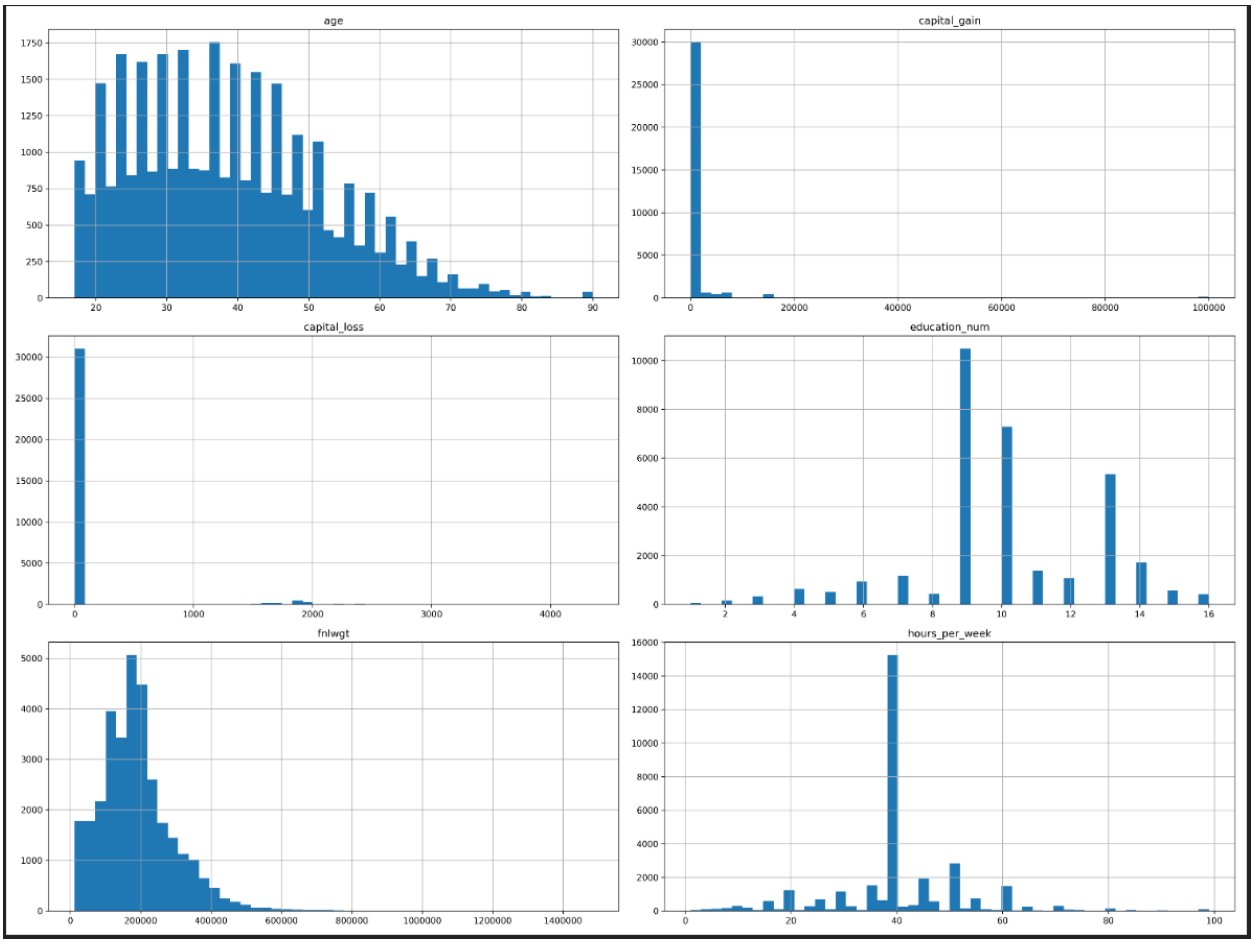
1) Age
2) Fnlwgt
3) Hours per week
4) Capital_gain
5) Capital_loss

Dimension attributes

1) workclass
2) education (education_num is the encoding for this)
3) occupation
4) relationship
5) race
6) sex
7) native_country
8) economic_indicator
9)

DATASET: Census

Attributes Histogram plot. This is the distribution of the data



We do an exhaustive search over the combinations of (group by attribute 'a', measure 'f', aggregate function f) and find the combinations that maximize the distance between the distribution of the target group (married people) and the distribution of the reference group (unmarried people). We intelligently merge and batch queries, reducing the number of queries issued to the database minimizing the number of scans of the data since the queries only differ in attributes used for grouping and aggregation. Regarding aggregation we stick to using only **avg, sum, min, max and count.**

Q can select any subset of records from the Census table resulting in D_Q. Visualization $V_?$ can be translated into an aggregate over group-by query on the underlying data and is represented as a function represented by (a, m, f) where $a \in A, m \in M, f \in F$.

$V_?(D)$ represents the results of grouping the data in D by a, then aggregating the m values using f;

$V_?(D_Q)$ represents a similar visualization applied to the data in $D_?$which is compared to a reference dataset (called $D_?$) to see if it has high utility. The reference dataset $D_?$may be defined as the entire underlying dataset (D), the complement of $D_?(D - D_Q)$ or data selected by an arbitrary query, $D_{Q'}$The result of the view queries view queries are summaries with two columns, namely a and f(m). To ensure that all aggregate summaries have the same scale, we normalize each summary into a probability distribution (i.e. the values of f(m) sum to 1). Utility of $V_?$ is the distance between the probability distributions for target view ($P([V_?(D_Q)]$) and the reference view ($P([V_?(D_R)]$)

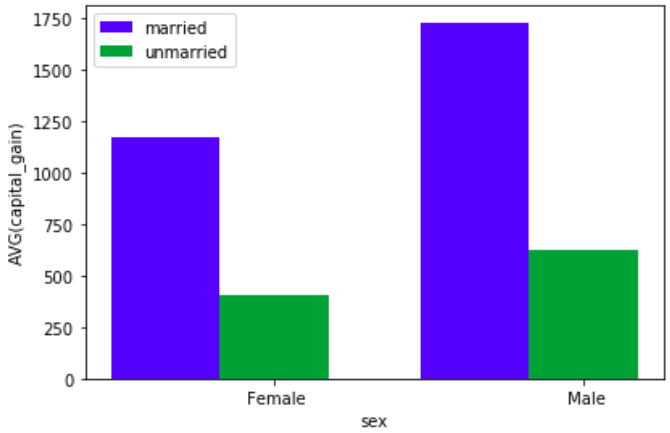
Methodology

A typical visualization $V_?$ can be translated into an aggregate over group-by query on the underlying data and is presented as a function of the triple (a, m, f) where $a \in A, m \in M, f \in F$. For example, in Figure 1, the analyst may build a chart showing capital gain group-by sex for the target data (married) that deviate from the reference data (unmarried).

The corresponding query is:

SELECT sex, avg(capital_gain) FROM married GROUP BY sex;

SELECT sex, avg(capital_gain) FROM unmarried GROUP BY sex;



CPU times: user 359 ms, sys: 29.1 ms, total: 388 ms

Wall time: 691 ms

However, the analyst often interested in the *top-k* visualizations instead of considering all $\mathbf{2 * a * m * f}$ **visualizations**, which takes roughly **41s** in our settings. This clearly does not meet the response time for an interactive analysis.

In this project, we replicate the results of *the phased-execution framework* which alternates between a sharing-based optimization engine (to update the partial results of the visualizations) and a pruning-based optimization engine (to drop low utility visualizations). We choose the number of phases to be 5.

Metric

We use the **Kullback-Leibler divergence (KLD)** as our scoring function. We treat the aggregate value as unnormalized probability of the corresponding categorical variable, according to the paper's definition of interesting distribution. We ignore the categories with missing value in either the target or the reference distribution. We assign score 0 to views with no data.

$$D_{KL}(P||Q) = \sum_i P(i) * \log\left(\frac{P(i)}{Q(i)}\right)$$

Sharing-based optimizations

We combine multiple aggregates and multiple GROUP-BYs by using the PostgreSQL GROUPING SETS command for sharing computation across different dimension attributes. Here is an example query

```
SELECT workclass, education, occupation, avg(age), sum(age), avg(fnlwgt), sum(fnlwgt)
FROM married GROUP BY GROUPING SETS ((workclass), (education), (occupation));
```

```
SELECT workclass, education, occupation, avg(age), sum(age), avg(fnlwgt), sum(fnlwgt)
FROM married GROUP BY GROUPING SETS ((workclass), (education), (occupation));
```

Total running time for the whole dataset: See section 2.2 in SeeDB Experiments.ipynb

Pruning-based optimizations

We implement two pruning schemes: **confidence interval-based** pruning and **multi-armed bandit** pruning.

For confidence interval-based pruning, we first sort the current views in consideration, by the upper bound of their utility scores. We then discard every views that have the utility score's upper bound less than the worst lower bound of the *top-k* views. We choose the 95% confidence interval (δ = 0.05).

- For each view, we maintain a running average μ by averaging the utility score (KL-D) has seen so far.
- Compute the confidence interval ε as in theorem 4.1, m is the iteration number , δ = 0.05, N= number of data partitions.

THEOREM 4.1. *Fix any $\delta > 0$. For $1 \leq m \leq N-1$, define*

$$\varepsilon_m = \sqrt{\frac{(1-\frac{m-1}{N})(2\log\log(m)+\log(\pi^2/3\delta))}{2m}}.$$

Then: $$\Pr\left[\exists m, 1 \leq m \leq N : \left|\frac{\sum_{i=1}^{m} Y_i}{m} - \mu\right| > \varepsilon_m\right] \leq \delta.$$

-
- The upper bound is $\mu + \varepsilon$, the lower bound is $\mu - \varepsilon$.
- The threshold to reject views is the smallest lower bound of the *top-k* views (rank by upper bound). View with upper bound smaller than threshold will be discarded.

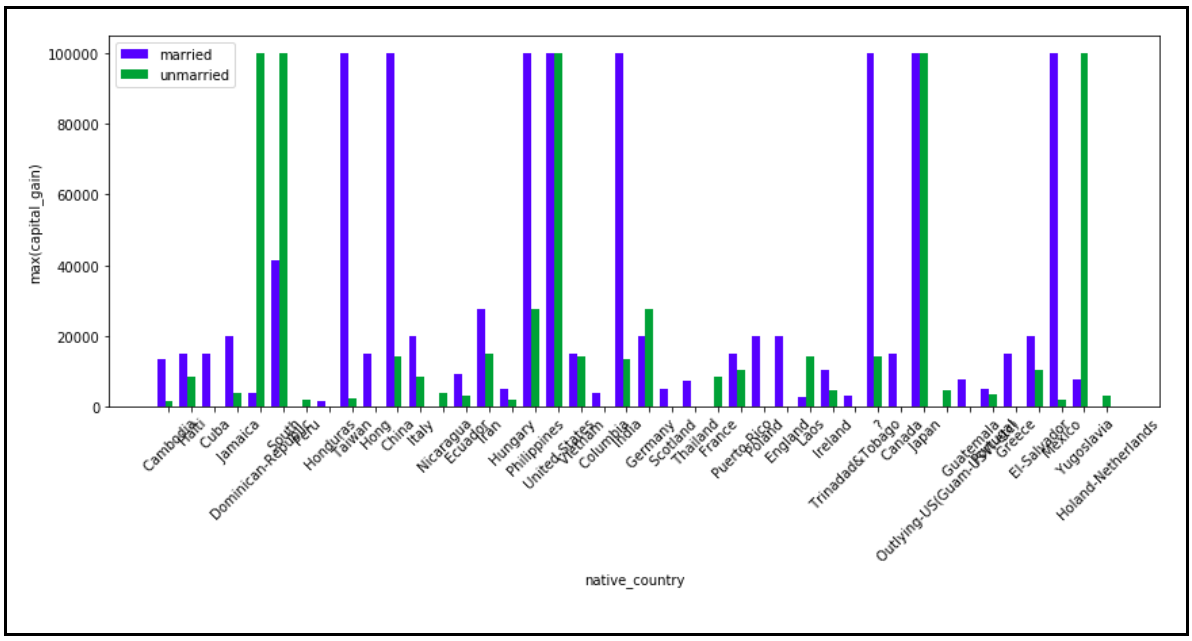
For multi-armed bandit pruning, we implement the Successive Accepts and Rejects algorithm as described in the paper. Firstly, we sort the views by the mean of their utility scores. Then, we compute Δ_1 by subtracting the highest mean to the $(k+1)^{th}$ highest mean and Δ_n by subtracting the k^{th} mean to the worst mean. Depending on the value of Δ_1 with respect to Δ_n , we either accept the highest score view or reject the lowest score view. Note that in both cases, we do not consider the selected view in the next phase. Thus, we constantly reduce the number of views being considered by 1 after each phase. We also stop when the *top-k* views are selected.

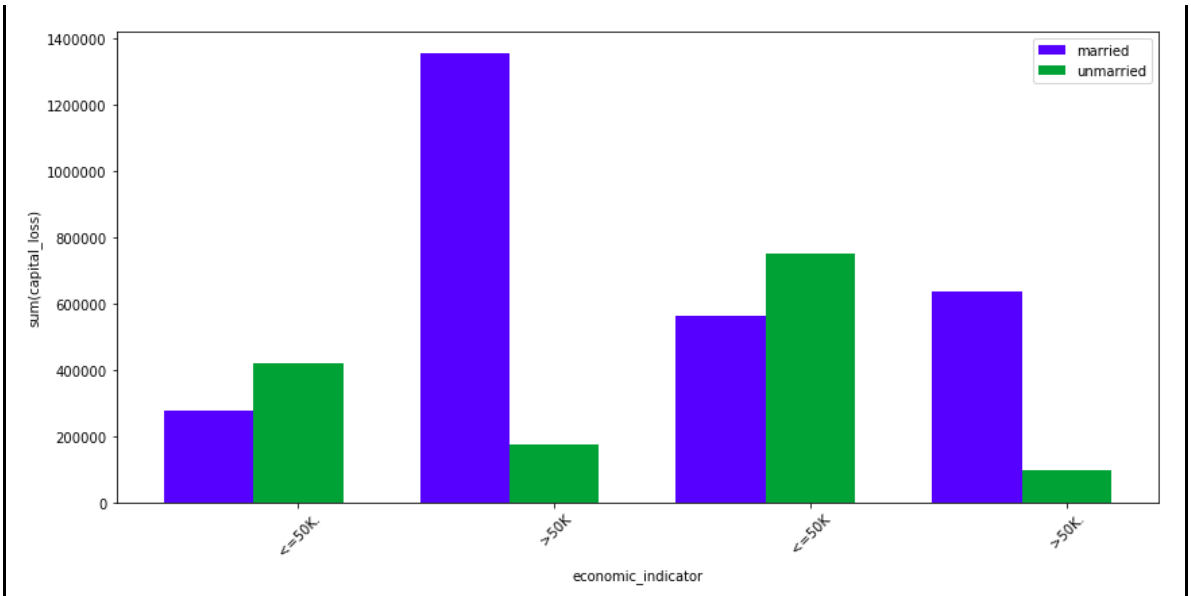
Visualizations

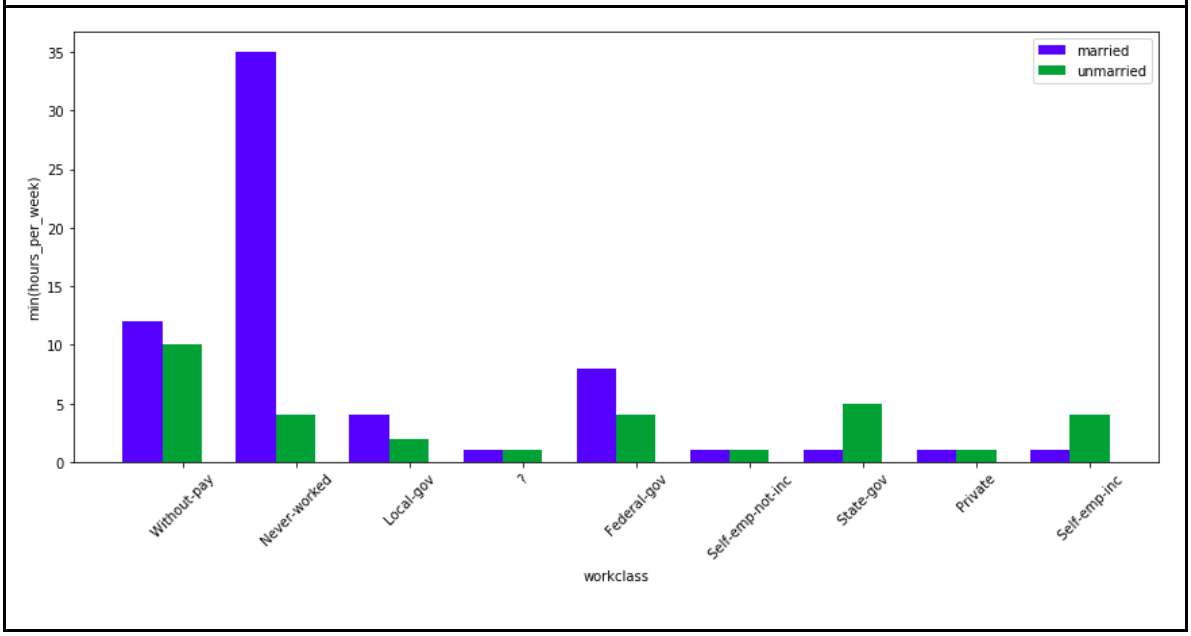
We visualize the *top-k* views found by our method and compare them with the *top-k* views from the exhaustive search along with the latency

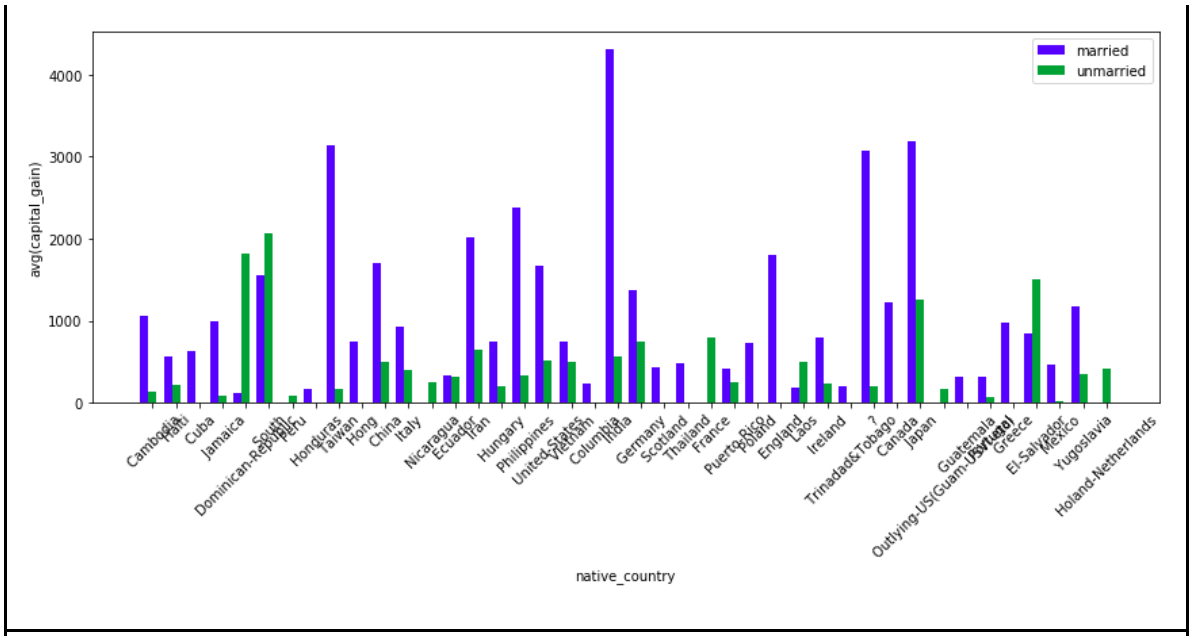
1. **TOP 5 views return by Shared Optimization**

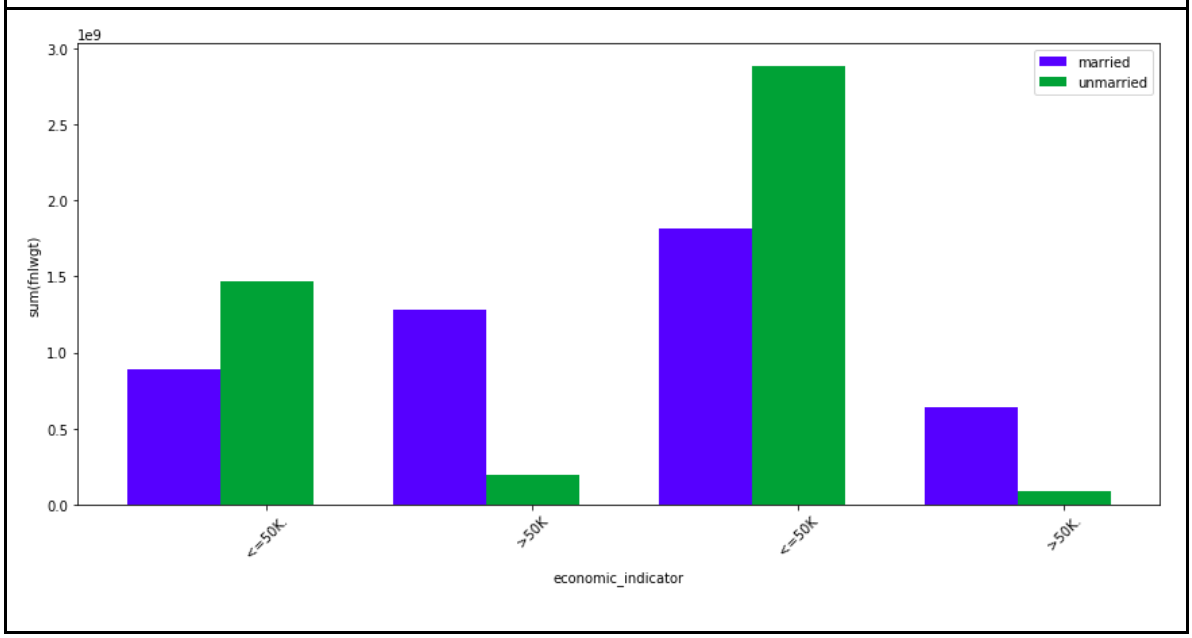
CPU times: user 3.65 s, sys: 127 ms, total: 3.77 s
Wall time: 6.4 s




1400000
1200000
1000000
800000
600000
400000
200000
0
sum(capital_loss)
married
unmarried
<=50K
>50K
<=50K
>50K
economic_indicator

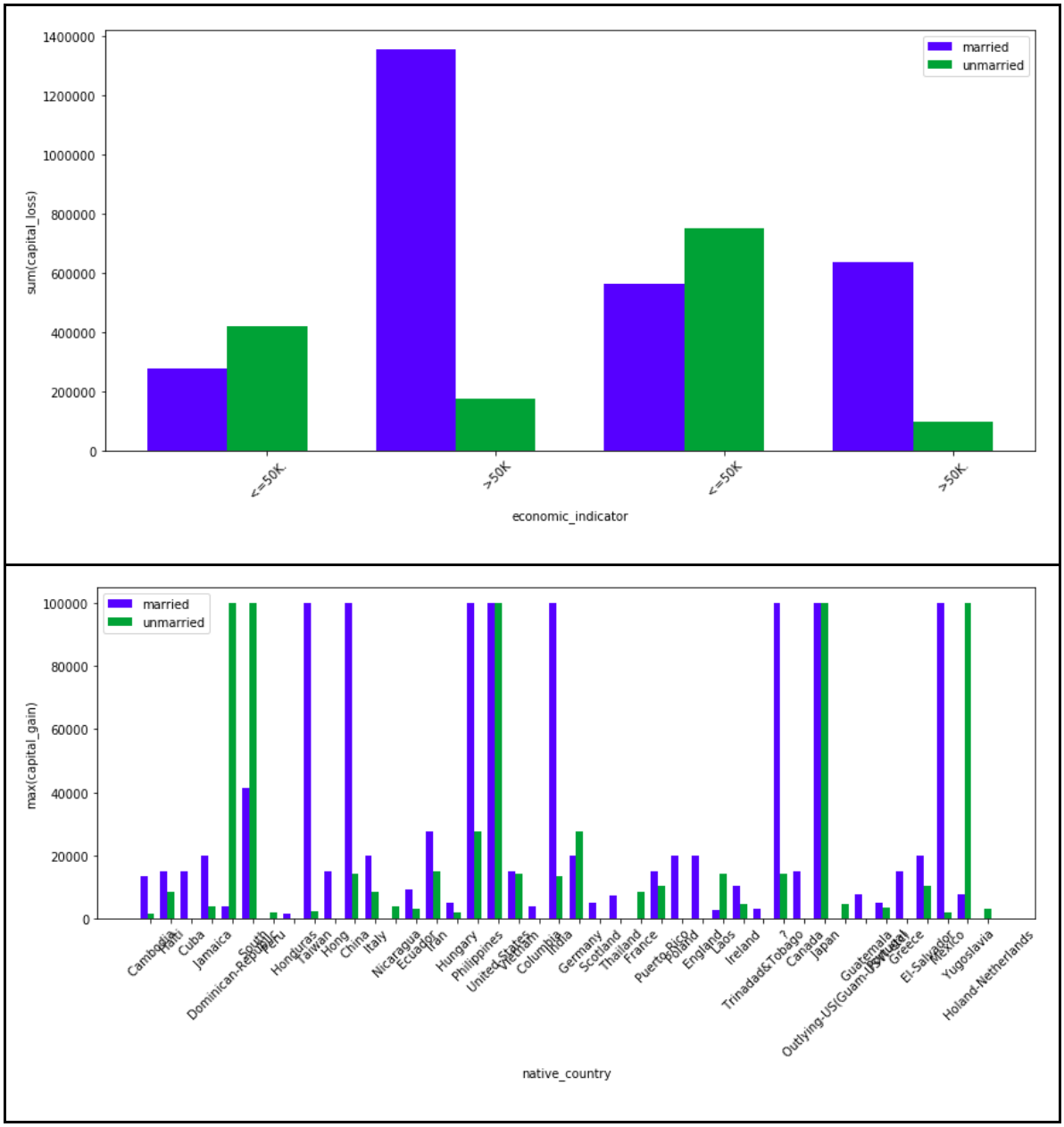

35
30
25
20
15
10
5
0
min(hours_per_week)
married
unmarried
Without-pay
Never-worked
Local-gov
?
Federal-gov
Self-emp-not-inc
State-gov
Private
Self-emp-inc
workclass

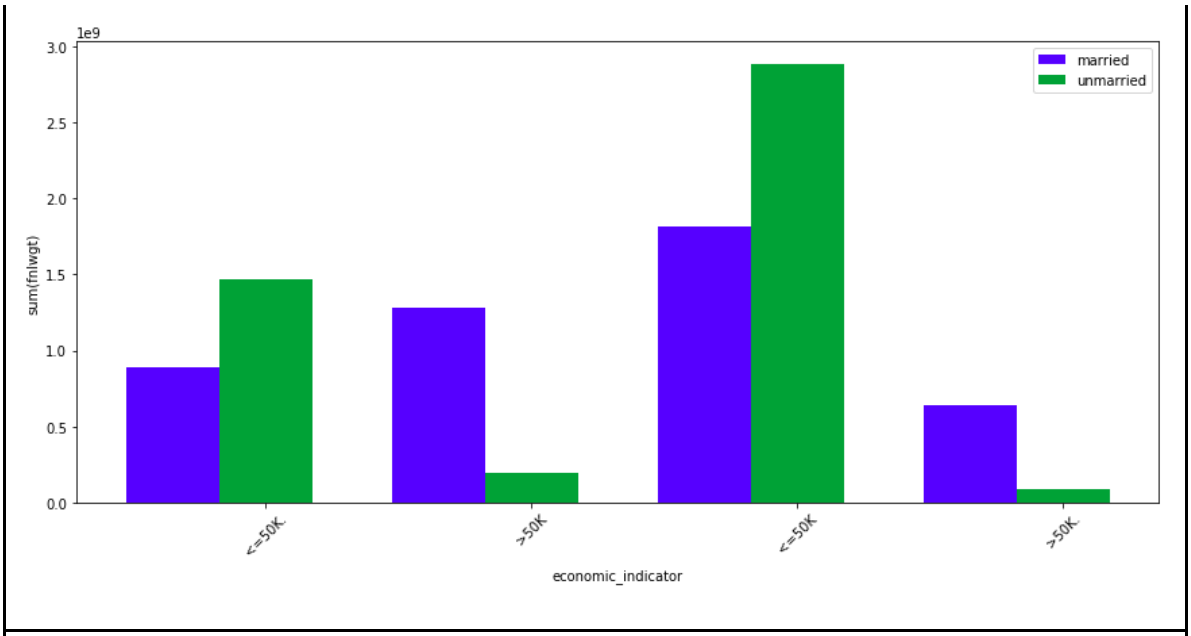

married
unmarried
avg(capital_gain)
4000
3000
2000
1000
0
native_country

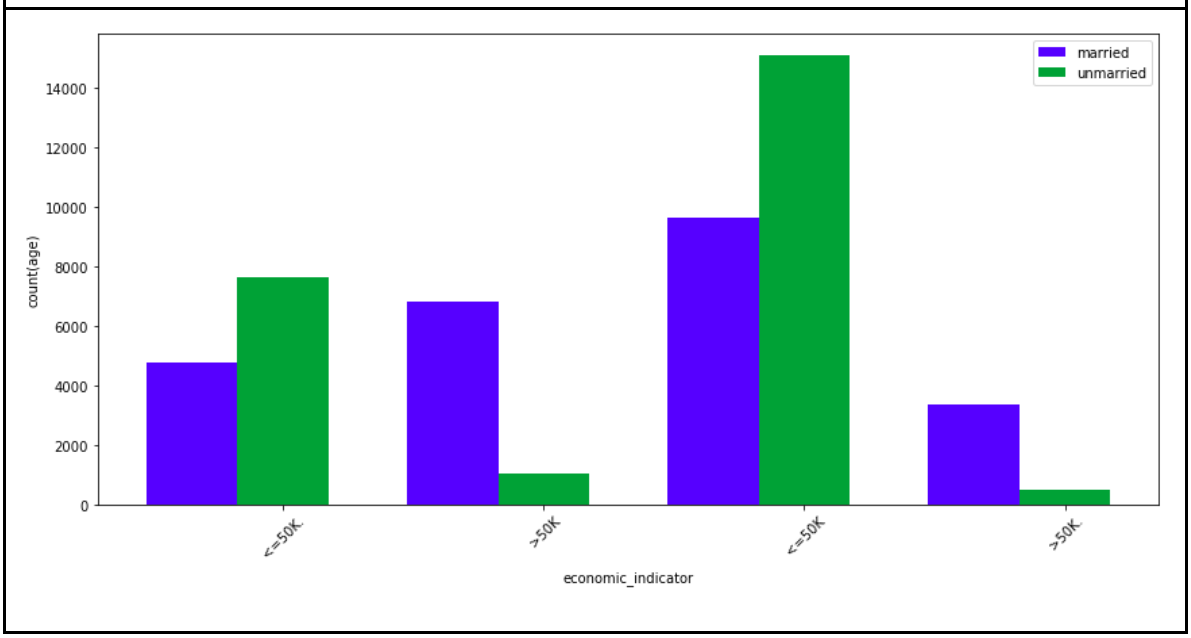

1e9
3.0
2.5
2.0
1.5
1.0
0.5
0.0
married
unmarried
sum(fnlwgt)
<=50K
>50K
<=50K
>50K
economic_indicator

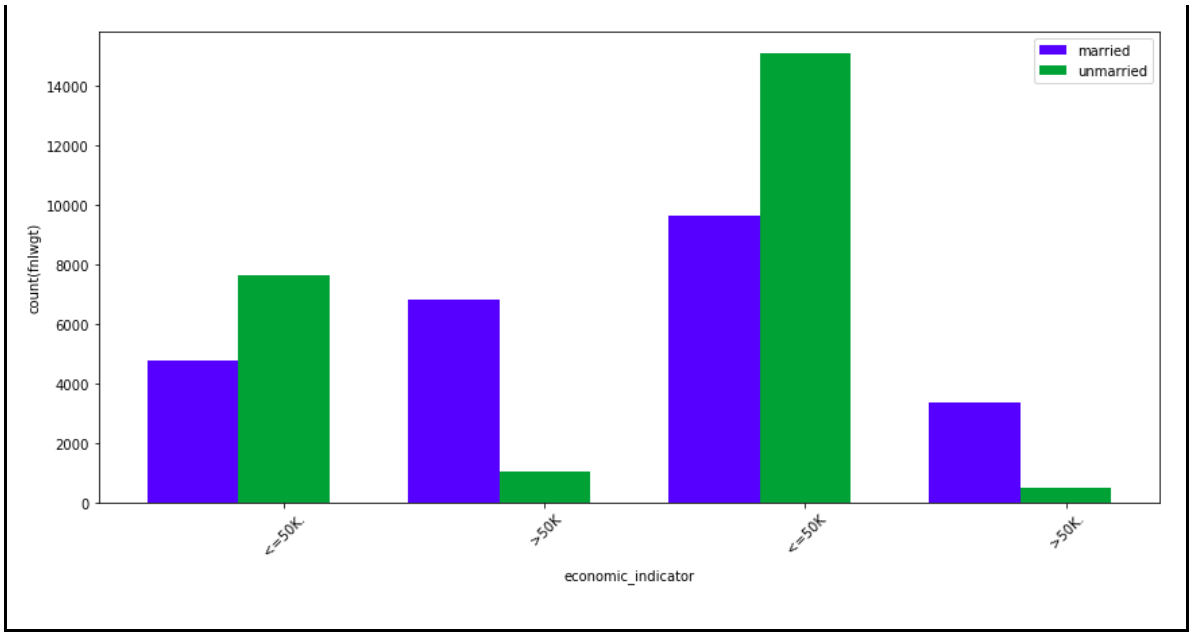
2. **TOP 5 views return by the proposed method (interval-based pruning)**

CPU times: user 7.28 s, sys: 138 ms, total: 7.42 s
Wall time: 9.11 s



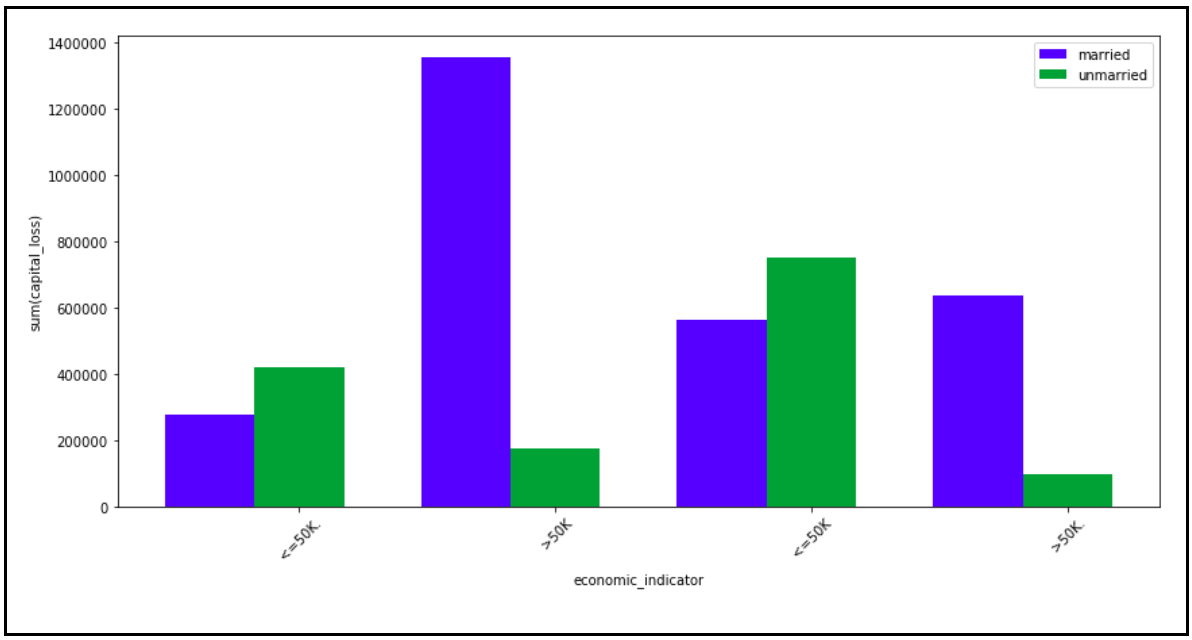

1e9
3.0
2.5
2.0
1.5
1.0
0.5
0.0
sum(fnlwgt)
married
unmarried
<=50K
>50K
<=50K
>50K
economic_indicator

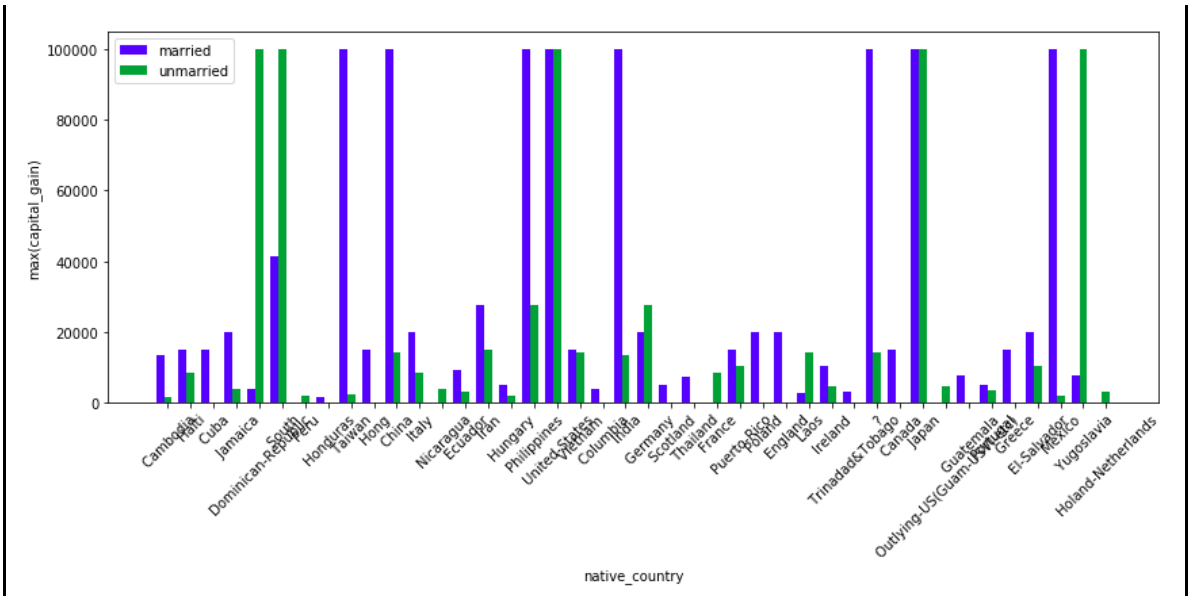

14000
12000
10000
8000
6000
4000
2000
0
count(age)
married
unmarried
<=50K
>50K
<=50K
>50K
economic_indicator

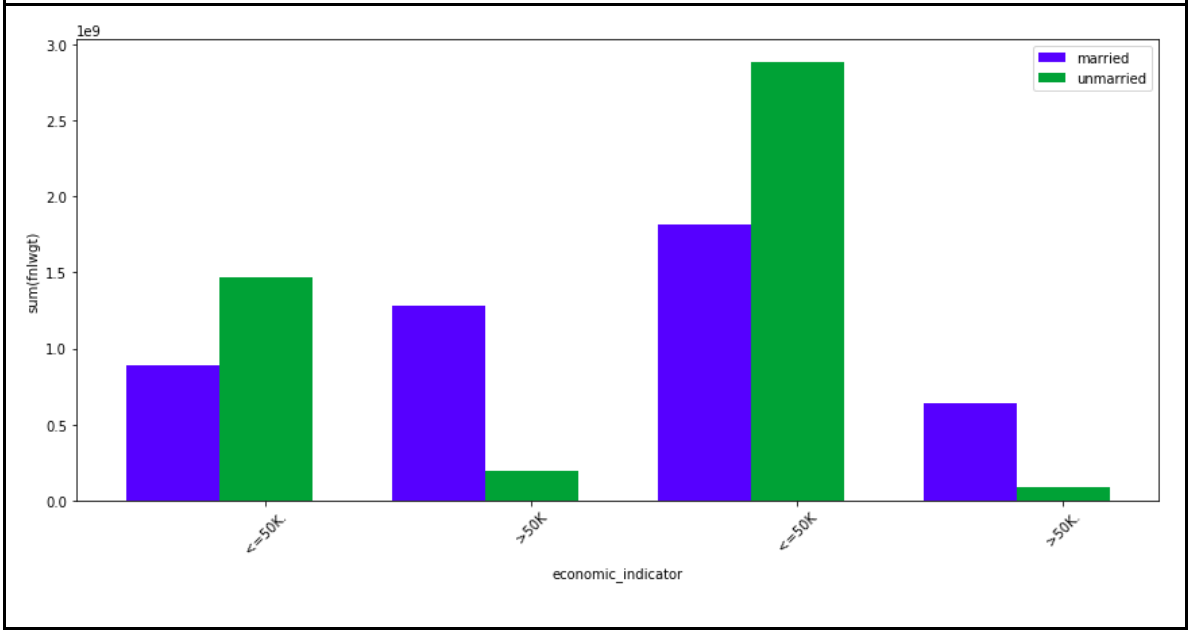


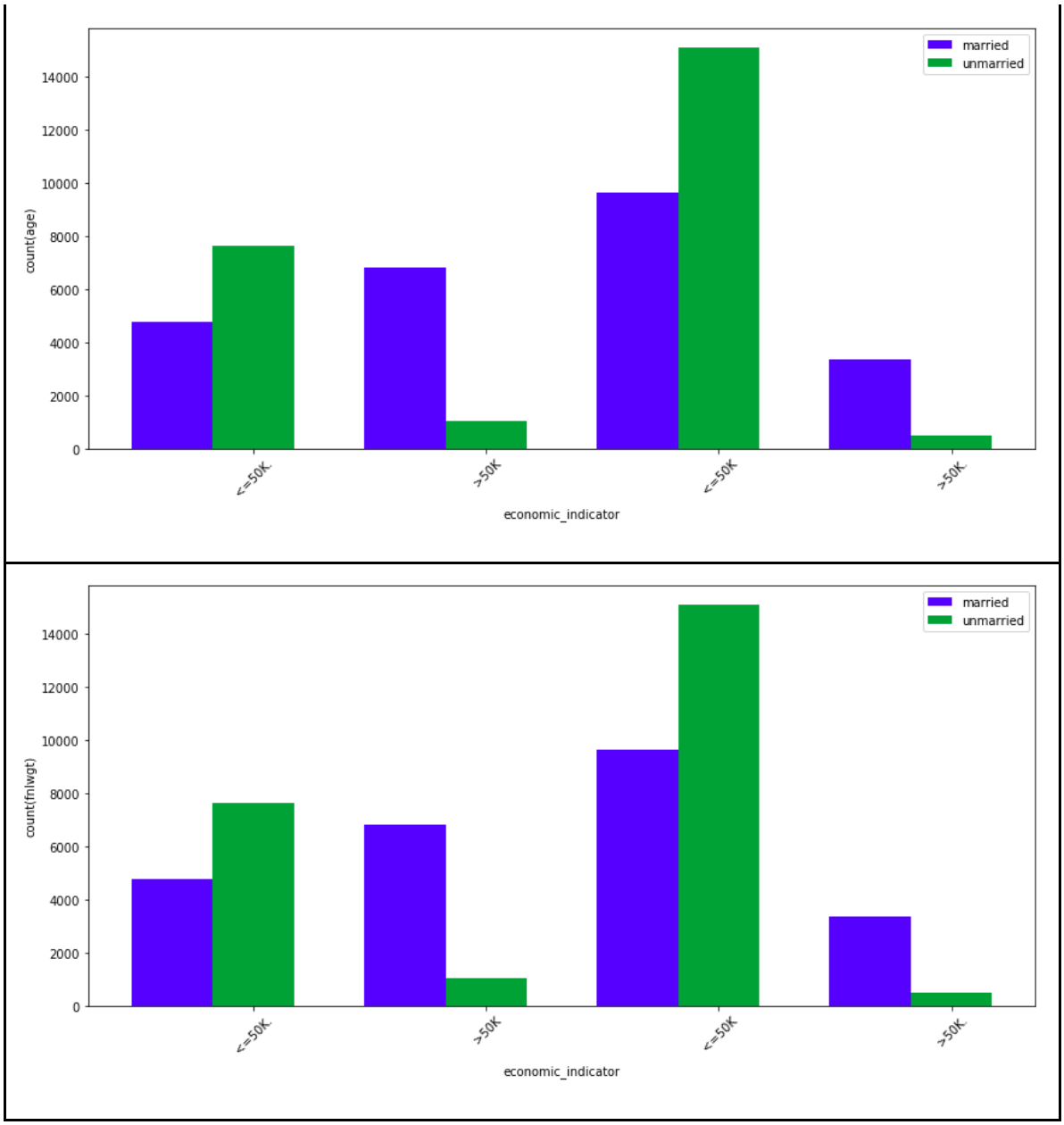
3. TOP 5 views return by the proposed method (MAB pruning)

CPU times: user 6.99 s, sys: 117 ms, total: 7.11 s
Wall time: 8.75 s




married
unmarried
max(capital_gain)
native_country


1e9
married
unmarried
sum(fnlwgt)
economic_indicator
<=50K
>50K
<=50K
>50K



Latency comparison

Method	User	Sys	Total time
Sharing-based Optimizations	3.65 s	127 ms	3.77 s
Interval-based pruning	7.28 s	138 ms	7.42 s
Multi-armed bandit pruning	6.99 s	117 ms	7.11 s

Conclusion

KL divergence works well as a deviation metric to capture the interestingness of a visualization. We found that not ignoring the categories with missing value in either the target or the reference distribution for the KL divergence metric gives visualizations with missing data (views involving relationship attribute) that may not be fruitful for further exploration. The high KL divergence values prune everything in the first phase and do not explore any useful visualizations. Our visualizations show much better recommendation criteria. The attributes and functions identified may be worth exploring further to infer interesting trends or patterns. Our implementation works at an interactive timescale, giving swift recommendations. We attempted both the pruning mechanisms mentioned in the original paper and compared our latencies against exhaustive search which shows almost a 10x speed up with sharing-based optimization and a 5-6X speed up in the pruning-based optimizations.

CODE:

The code for this project can be found on

https://github.com/snknitin/-SeeDB

------X------